Sensor Data Processing on a Reconfigurable Processor

Gregory W. Donohoe
ECE/CAMBR, POB 441023
University of Idaho
Moscow, Idaho 83843

Pen-Shu Yeh
NASA GSFC Code 567
Greenbelt MD 20771-0001

***Abstract*- This paper describes the implementation of two sensor data processing tasks on the Reconfigurable Data Path Processor (RDPP). These tasks are focal plane array readout correction, and Fourier transform hyperspectral imager data conversions. These tasks illustrate reconfigurability, dynamic data path switching, and data path parallelism. The two challenge problems were demonstrated on an RDPP simulator, helping to validate the RDPP architecture.**

I. Introduction

The Reconfigurable Data Path Processor (RDPP) is a new on-board data processing architecture for spacecraft instrument data processing, developed under ESTO sponsorship. The RDPP is run-time reconfigurable, that is, its internal architecture is rewired under software control to optimize it for the processing task at hand. The RDPP is one of the very few processors designed specifically for reconfigurable computing.

The RDPP contains 16 on-board processing elements, each equipped with a 24-bit multiplier, an arithmetic-logic unit (ALU), signal conditioning logic, and switching logic [1, 2]. Programmable interconnects enable the processing elements to be connected in a synchronous data flow pipleline. Five 24-bit input and output ports provide a data interface, and 10 control signals synchronize the RDPP to external hardware. An on-board execution unit with 256 words of memory synchronizes the firing of processing elements. A byte-oriented host interface connects the RDPP to a host processor, which can be a microcontroller or a remote processor connected over a digital bus. The RDPP project has also produced a suite of development software, including application development tools and a functional simulator.

Fig. 1 shows an RDPP processing pipeline. The processing elements (PEs) can perform arithmetic and logical operations, and also serve as run-time switches to join two data path segments. Each PE and IO module has a registered input with an independent fire signal, controlled by the on-board execution unit. In the figure, τ represents a processing delay.

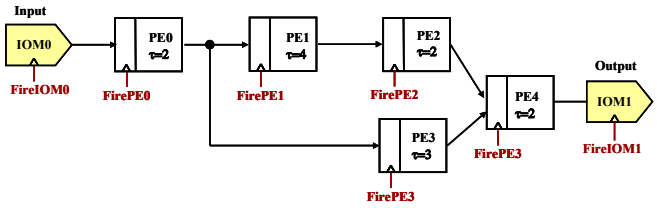

Figure 1: RDPP Processing Pipeline

Each RDPP chip contributes 16 processing elements to a processing pipeline. By tiling RDPP chips, we can construct a pipeline of arbitrary size.

RDPP operation occurs in two phases:

1. Configuration
2. Execution

In the configuration phase, a host processor sets up the function of each module and establishes the topology of the pipeline. In execution, the on-board execution unit "fires" the executable modules, i.e., the PEs and IO ports. A module fires when new data are latched into its input.

II. Challenge Problems

The RDPP is demonstrated on two challenge problems: Focal Plane Array Sensor Readout Correction, and Fourier Transform Hyperspectral Imager Data conversion. These tasks were demonstrated using RDPP standalone simulator, RDPPSim.

A. Focal Plane Array Readout Correction

A solid-state focal plane imaging array (FPA) consists of a matrix of sensors. Each sensor has a photodetector, typically a photodiode; a charge storage mechanism; and a small current-to-voltage conversion amplifier. An analog readout mechanism such as a charge coupled device (CCD) pipeline enables the voltage at each pixel to be multiplexed to an analog-to-digital converter (ADC). The imaging elements, or

pixels, are not all uniform. Due to process variations, each pixel has a unique offset and a unique gain error. This results in so-called "fixed pattern noise" in an image. FPA readout correction consists of reading a pixel, multiplying by a gain correction factor, and subtracting an offset correction factor.

In addition to nonuniformity, a pixel may be "hot" or "dead". A pixel becomes hot when an ion, due to cosmic radiation, is trapped in the photodiode and causes a constant current to flow, producing a bright spot in the image. A dead pixel is disabled due to an electronic fault. Hot and dead pixels are not responsive to changes in illumination. We can approximate a complete image by replacing each bad pixel with a spatial average of neighbors.

B. *Hyperspectral Imager Data Conversion*

The purpose of Fourier Transform Hyperspectral Imager (FTHSI) data conversion is to convert the raw signal data from a FTHSI instrument into a spectral intensity signal. This requires appodizing the raw signal, and computing the magnitude Fourier transform of the signal. The data used in this example come from a Kestrel FTHSI instrument similar to one flown on the Air Force Mighty Sat I.2 satellite [3, 4].

Fig 2 shows a block diagram of the FTHSI instrument in a pushbroom arrangement. A cylindrical lens system images a stripe of the scene through a slit aperture, to a Sagnac interferometer. The split-beam interferometer produces interferograms, optical images that contain the Fourier transform of the spectral intensity. An imaging array captures these interferograms, which are digitized. The data conversion block is a digital processor, which appodizes each interferogram to remove artifacts, then computes the magnitude of the Discrete Fourier Transform (DFT). This produces the spectral intensity at a point in the image.

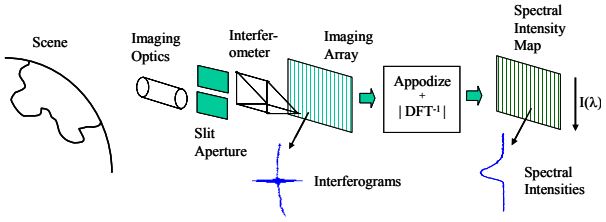

Figure 23: Fourier transform hyperspectral imager schematic diagram.

The operations required are:

1. Remove artifacts due to vignetting.
2. Apply a Hamming window.
3. Compute the magnitude DFT

III. PROCESSING ON THE RDPP

We are challenged to formulate these two processing problems for execution on a reconfigurable processor.

A. *FPA Readout Correction on the RDPP*

Focal Plane Array readout correction demonstrates two important features of the RDPP: 1) reconfigurability to adapt to different phases of a problem, and 2) run-time conditional data path selection. Readout correction consists of two disjoint operations:

1. For good pixels, multiply by a gain correction factor, and subtract and offset correction.
2. For bad pixels, replace the pixel with a spatial average of neighbors.

First, we must acquire the correction data, through calibration. This requires several configurations. Fig. 3 shows the hardware configuration. In addition to the RDPP, we need two read only memories (SRAMs) to hold gain and offset corrections, and a First-In, First-Out (FIFO) line buffer, to delay one line of the image. (Addressing logic for the SRAMs is not shown.)

1) *First configuration: acquire offset data*. We close the aperture to block all light, acquire an image, and store it in memory. This is the offset or dark current error, scaled by the corresponding gain error.

2) *Second configuration: acquire gain error and find bad pixels*. We present a known, constant gray scene, acquire and store an image. Ideally, this image will be uniformly gray. In fact, it will exhibit fixed pattern noise due to the offset and gain errors. From this, we can easily derive the gain error.

We can now compare the two images to find pixels that

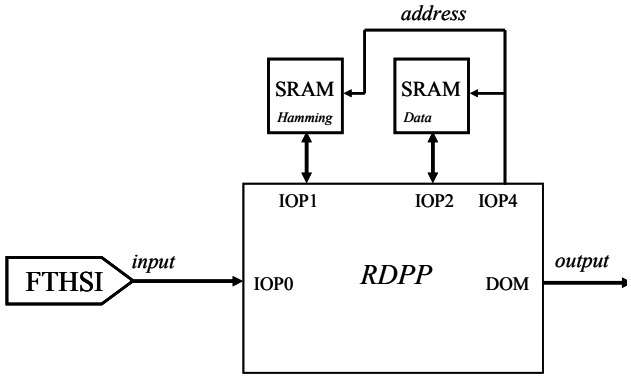

Figure 32: hardware configuration for FPA readout correction.

have not changed. These are dead pixels; we label them by setting an unused high-order bit in the offset memory.

3) *Third configuration: compute the reciprocal gain* function. Gain correction requires dividing each pixel by a gain correction factor, or equivalently, multiplying by the reciprocal. Division is a complex operation, however, and the RDPP does not have a hardware divider. However, we can configure the RDPP to form a pipeline for division. We read each gain factor from memory, pass it through this pipeline to compute the reciprocal, and write the reciprocal back into memory. During the execution phase, only multiplication is required.

We use a well-known recursive division algorithm. For more details, see [5].

4) Fourth configuration: execution. We now have the calibration data stored in two separate memories, consisting of:

1. The offset values, tagged with a special code for "bad pixels"
2. The reciprocal of the gain error.

Now we can read and correct images. In our implementation, the RDPP executes the two types of correction – nonuniformity correction, and bad pixel replacement – simultaneously, and selects the correct value to output, depending on whether the pixel has been tagged as "bad".

The pipeline is shown in Fig. 4. OIP0 through IOP3 are input-output ports, configured for input. DOP is a dedicated output module. After a latency of 9 cycles, we obtain a corrected pixel on every instruction cycle.

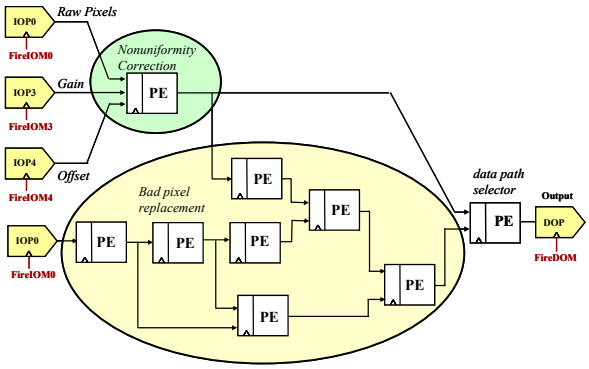

Figure 4: RDPP pipeline for pixel readout correction.

B. *FTHSI Data Conversion on the RDPP*

Data arrive from the FTHSI instrument as raw interferograms, like the one shown in Fig. 5. The information is concentrated in the "burst" at the center. The vertical units are simply analog-to-digital converter counts, as this signal has not been calibrated. The signal also has artifacts, which must be removed through a process called appodization. The signal has been digitized to 12 bits from a focal plane imaging array.

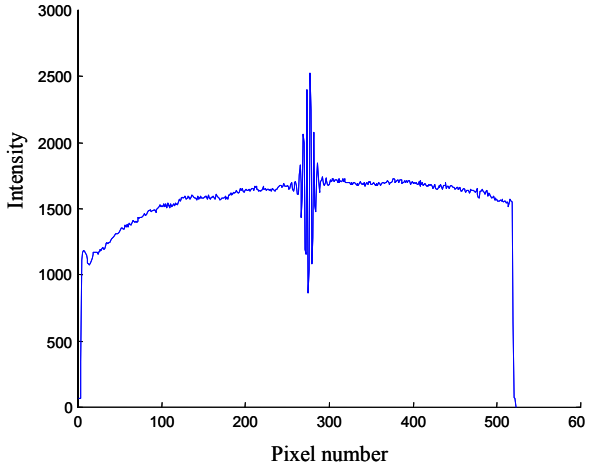

Figure 5: raw interferogram.

The information-bearing signal rides on an underlying curve, which is due to vignetting of the optical system, as are the spikes at the ends. We must clean up the signal and remove the mean. To achieve this, we pass the signal through a high pass filter. At the same time, we prepare the signal for the Discrete Fourier Transform, multiplying the signal point-by-point by a Hamming window. A well-known method for reducing windowing effects, the Hamming window is simply one half of a period of a cosine function fitted to the width of the curve. This is computed offline and stored in memory. Fig. 4 shows the signal after appodization and application of the Hamming window.

Next, we compute the magnitude Fast Fourier Transform on the signal. This generates the magnitude spectrum of Fig 5. This signal has not been calibrated for wavelength or intentisy, so the wavelength and spectral intensity axes are unitless.

Computing the Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT) on the RDPP requires a different

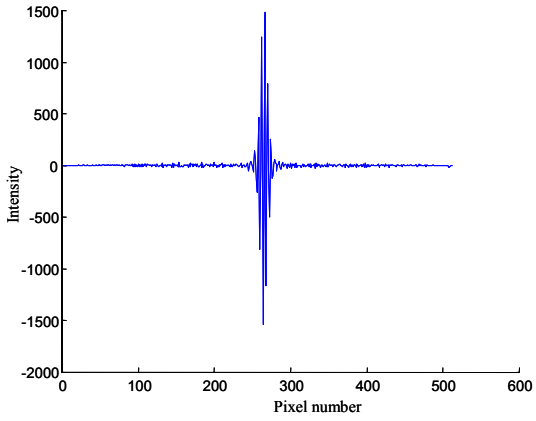

Figure 6: Appodized interferograms.

approach than the familiar Cooley-Tukey FFT algorithms. The Cooley-Tukey algorithm is optimized for a random-access computers with a single multiplier. It employs a divide-and-conquer technique to significantly reduce the number of multiply operations. This requires frequent re-ordering of the data, however. In contrast, the RDPP is a multiplier-rich computational platform, and is not random-access: it has no pool of randomly accessible memory. Data shuffling on the RDPP is not easy. Therefore, the optimal FFT algorithm for the RDPP is not the classical one, but is based instead on the Goertzel algorithm. This is developed in detail in [6].

Fig. 7 shows a converted spectrum. The signal has not been spectrally calibrated, so the scales on the axes are meaningless. However, the form of the spectrum is sensible. The target is an optical flat, and the source is an incandescent lamp; the form of the spectrum is a Planck's Law curve, which we expect.

Figure 8 shows a possible hardware configuration for FHSI data conversion. The memory on the left holds the pre-computed Hamming window coefficients, and the memory on the right holds the data being converted. In this example, I/O port IOP4 provides memory addresses to both memory chips, generated in RDPP software.

One RDPP can compute eight magnitude Fourier coefficients simultaneously, with a single read through the data. The magnitude is computed directly – we don't have to compute the complex-valued DFT first, then take the magnitude of that. The significant data are contained in about 128 data points in the center of the interferogram signal. With the Goertzel technique, we can compute only the coefficients we need, unlike block-oriented FFT algorithms, where we must compute the FFT on an entire sequence, and extract the useful portion. We can convert 128 data points with a single RDPP chip in seven blocks of eight points each. This requires

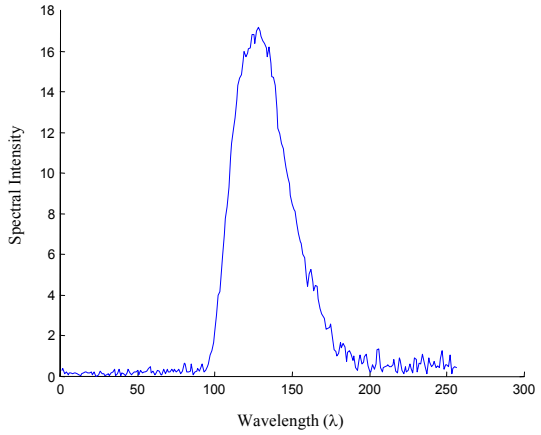

Figure 7: Converted spectral intensity signal.

se

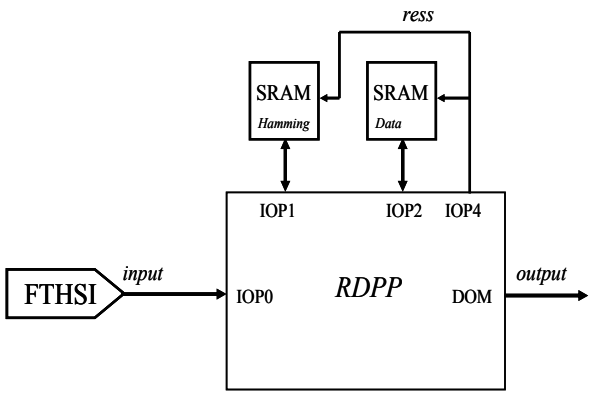

Figure8: A hardware configuration for FTHSI data conversion.

IV. PERFORMANCE

1) *Readout correction.* Our implementation of the Focal Plane Array pixel readout correction requires four configurations: three to acquire and store calibration data, and one to correct images. The image correction phase has a latency of 9 instruction cycles, and a through put of one processed pixel per instruction cycle. At an anticipated cycle rate of 60 MHz, this is 60 megagpixels per second.

This implementation requires two random access memories and a FIFO line buffer.

2) *Hyperspectral Imager Data Conversion*. Appodization can be performed on a single FTHSI interferograms in a single pass through the RDPP. The Hamming window can be applied at the same time. The appodized and windowed image (128 points) is stored in memory.

Computing the magnitude inverse transform on the 128 point sequence in one RDPP takes seven passes through the data. Between passes, the host must load new coefficients into 8 processing elements. Each pass though the data takes about 2.1 microseconds. Loading the 8 coefficients takes about 0.8 microseconds. Thus, converting the 128-point sequence with one RDPP chip requires $(2.1+0.8)\times 7 = 20.3$ microseconds. This requires one RDPP chip and at least one external memory; each memory holds only 128 points of data plus 128 points of Hamming window.

Since Goertzel algorithm computes the Fourier coefficients independently, we can use multiple RDPP chips and compute the FFT on the entire sequence at once, with one read through the data, and no need to re-load coefficients, in about 2.1 microseconds. By combining appodization and FFT computation, a data rate of 2.8 megasamples per second is possible.

Conclusions

We have demonstrated two challenge problems on the Reconfigurable Data Path Processor. Both are typical spacecraft instrument data processing problems. Both problems were demonstrated using the RDPP simulator, which produces results that are functionally equivalent to the chip itself. These demonstrations represent an important step in the validation of the RDPP architecture.

Acknowledgements

The authors wish to acknowledge the contributions of the RDPP development team: David Buehler, Enrique Coen-Alfaro, Jagdish Sabde, Nilesh Vyas, Mrinal Kochar, and John Purviance.

References

[1] G.W.Donohoe and P.-S. Yeh, "Reconfigurable Data Path Processor", *Proc. NASA Earth Sciences Technology Conference*, University of Maryland, August 29, 2001.

[1] Donohoe, G.W. and P.-S. Yeh, "A low power reconfigurable processor", *Proc. IEEE Aerospace Conference*, Big Sky, MT, March 9-16, 2002.

[3] Otten, L. John III, Eugene W. Butler, J. Bruce Rafert, R. Glenn Sellar, "The design of an airborne Fourier transform visible hyperspectral imaging system for light aircraft environmental remote sensing", *Imaging Spectrometry*, SPIE Vol. 2480, April 1995.

[4] Otten, L. John III, Andrew D. Meigs, R. Glenn Sellar, "Calibration and performance of the airborne Fourier transform visible hyperspectral imager (FTHSI)", *Proc. Second International Airborne Remote Sensing Conference and Exhibition*, San Francisco, CA, 24-27 June, 1996.

[5] Sabde, J., D. Buehler, and G. Donohoe, "Focal Plane Array Sensor Readout Correction on a Reconfigurable Processor", *Proc. 11th NASA Symposium on VLSI Design*, Coeur d'Alene, ID, May 38-29, 2003.

[6] Donohoe, G.W., P.-S. Yeh, and J. Purviance, The Fast Fourier Transform on a Reconfigurable Processor, *Proc NASA Earth Sciences Technology Conference*, Pasadena, CA, June 11-13, 2002.